VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Gentzenův důkazový kalkul

Gentzen Proof Calculus

2017 Bc. Jiří Spáčil

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jiří Spáčil**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Gentzenův důkazový kalkul
Gentzen Proof Calculus

Jazyk vypracování: čeština

Zásady pro vypracování:

Student vypracuje aplikaci, která bude sloužit jako pomůcka pro studenty logiky.
Cílem práce je automatizované dokazování platnosti úsudků nebo logické pravdivosti formulí predikátové logiky 1. řádu v sekventovém Gentzenově kalkulu.

Program bude umožňovat:
1. Jednoduchou práci s logickými formulemi.
2. Ekvivalentní úpravy formulí.
3. Aplikaci pravidel důkazového kalkulu.
Tyto práce bude provádět uživatel, program bude poskytovat účinnou asistenci, tj. kontrolu chyb, vodítko ke správnému řešení, případně i návrh správného řešení.

Seznam doporučené odborné literatury:

[1] Duží M. (2012): Logika pro informatiky a příbuzné obory. VŠB-Technická universita Ostrava. ISBN 978-80-248-2662-2. Dostupné online na http://www.cs.vsb.cz/duzi/Matlogika_ESF_Definite.pdf
[2] Halpern, J.Y., Vardi, M.Y. (1991): Model checking vs. theorem proving: a manifesto. In Artificial Intelligence and Mathematical Theory of Computation, V. Lisfchitz (ed.), Academic Press, pp. 151-176. Dostupné na http://www.cs.cornell.edu/home/halpern/papers/manifesto.pdf

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **prof. RNDr. Marie Duží, CSc.**

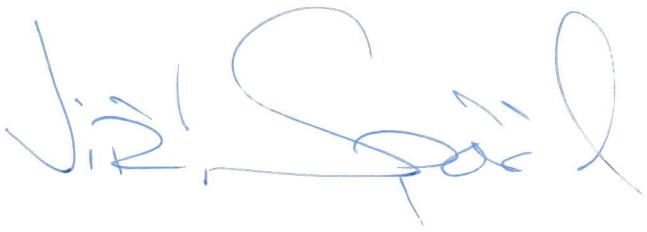
Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.



V Olomouci 1. dubna 2017

Rád bych poděkoval prof. RNDr. Marii Duží, CSc. za vedení diplomové práce, za její cenné rady a vstřícnost při konzultacích.

Abstrakt

Diplomová práce se zabývá studií Gentzenova (sekventového) kalkulu a jeho implementací do softwarového nástroje, který umí automatizovaně dokazovat formule výrokové i predikátové logiky prvního řádu. Cílem této práce je srozumitelně popsat funkci sekventového kalkulu a představit vše na konkrétních příkladech. Hlavní přínos je potom samotné sestavení univerzálního algoritmu pro automatizované dokazování a jeho implementace na platformě Windows. Práce představuje vytvořený software pro podporu výuky logiky a přesně popisuje jeho možné užívání.

Klíčová slova: důkazový kalkul, logický kalkul, sekventový kalkul, Gentzen, automatizované dokazování

Abstract

The thesis deals with study of the Gentzen's sequent calculus and its implementation in the software tool that can automatically prove propositional and first-order predicate logic formulas. The aim of this thesis is to clearly describe the functions of the sequent calculus and to demonstrate particular proofs by examples. The main contribution is a universal algorithm a universal algorithm for automatic theorem proving and its implementation on the Windows platform. The thesis presents the created software that will serve to support teaching logic and describes accurately its possible usage.

Key words: proof calculus, logic calculus, sequent calculus, Gentzen, automatic theorem proving

Obsah

Seznam obrázků

1 Úvod

1.1 Od základů logiky k důkazovým kalkulům

Logika je vědou o správném usuzování. V logice se snažíme rozlišit platné argumenty od neplatných, a odvodit logické důsledky z daných předpokladů. Navíc chceme tyto důkazové postupy formalizovat a automatizovat.

1.2 Trochu historie

Logika, jako vědní disciplína, se vyvíjela během staletí a vyvíjely se i její cíle. Za zakladatele logiky je považován řecký filosof Aristotelés. Od svých počátků byly otázky logiky směrovány na podstatu lidského usuzování a byly tedy svou povahou hlavně filozofické. Odtud pochází označení filozofická logika. Na přelomu 19. a 20. století došlo k výrazné formalizaci logiky. Šlo především o formalizaci usuzování v matematice a v jiných vědách, filozofické aspekty pak byly poněkud potlačeny. V současnosti neexistuje žádná zřetelná hranice mezi formální a filozofickou logikou. Mnohé aspekty usuzování zkoumané filozofickou logikou se podařilo formalizovat a zkoumat čistě matematickými metodami. Na druhou stranu, formální logika dokázala odpovědět na některé zásadní otázky, kterými se zabývali filozofové již od starověku. [1]

1.3 Logika a informatika

Vztah logiky a informatiky je velmi těsný. Logika je důležitá v informatice (formální metody specifikace, verifikace programů, metody analýzy dat) a elektrotechnice (logika elektrických obvodů). Logika se zabývá například algoritmickými aspekty usuzování a konstrukcí automatických dokazovacích systémů, které jsou schopny, byť omezeně, mechanicky odvozovat tvrzení z jiných. [1]

1.4 Důkazové kalkuly

Koncem 19. století se začala vyvíjet metoda důkazových kalkulů a to proto, že se v matematice začaly objevovat paradoxy, které byly spojeny s předpokladem aktuálního nekonečna. Tehdy byl David Hilbert(německý matematik) jedním z prvních matematiků, který se začal zajímat o tuto tématiku a vyhlásil program pro formalizaci matematiky, jehož cílem bylo mimo jiné, jak zabránit paradoxům. Dnes je tento program znám pod názvem Hilbertův program. [2]

1.5 Hilbertův program

Pojmem Hilbertův program se označuje snaha o formalizaci matematiky, až na úroveň jednoduchých axiomů, ze kterých by se daly korektně dokázat všechny matematické věty. Smyslem programu bylo redukovat složité matematické teorie (například matematickou analýzu) na jednoduché formální systémy a ty potom na jednoduchou aritmetiku, o které by se ukázalo, že je bezesporná a úplná. [2]

Hilbert vyhlásil tento program ve 20. letech 20. století, ale již v roce 1931 dokázal Kurt Gödel své věty o neúplnosti. Jejich důsledkem je fakt, že v teorii aritmetiky nelze dokázat bezespornost a úplnost aritmetiky samotné. Žádná teorie, která by se dala použít k popisu všech matematických pravd, nemůže dokázat svoji vlastní bezespornost. Hilbertův program je tedy neuskutečnitelný. [2]

1.6 Logické systémy

1.6.1 Výroková logika

Výroková logika (klasická logika) zkoumá usuzování o výrocích. Jednotlivé výroky nabývají pravdivostní hodnoty: Pravda, nepravda. Atomické výroky jsou nedělitelné. Např. "Prší". Ty se mohou spojovat pomocí (klasických) logických spojek: "a, nebo, pokud, právě když...". Slovní konstrukce, kterými se výroky spojují, budeme označovat symboly výrokových spojek. Binární spojky se vztahují vždy ke dvěma výrokům. Unární spojky se vztahují jenom k jednomu výroku. [3]

1.6.2 Predikátová logika

V matematice a logice se pojmem predikátová logika označuje formální odvozovací systém, používaný k popisu matematických teorií a vět. Predikátová logika je rozšířením výrokové logiky. Na rozdíl od výrokové logiky má bohatší vyjadřovací schopnost. Predikátová logika si všímá struktury vět. V každé větě rozlišuje individua, o kterých se něco predikuje. Predikát je chápán jako vlastnost nebo vztah. Teorémy výrokové logiky platí i v rámci predikátové logiky. Do výrokové logiky přidává kvantifikátory a možnost vyjádřit vlastnosti individuí a vztahy mezi individui. Individuum je prvek z nějaké množiny (univerza) a predikát je interpretován jako relace na této množině. Predikátové logiky máme dále rozděleny podle jejich expresivní síly na predikátové logiky n řádů. [1]

1.6.3 Fuzzy logika

Zabývá se tvrzeními, které mohou mít kromě pravdivostních hodnot pravda a nepravda i jiné hodnoty. Např. tvrzení „Zákazník je spokojený.“ může mít pravdivostní hodnotu 1 (pravda), ale i 0.8, pokud je spokojený ne zcela, nebo 0 (nepravda), pokud je nespokojený. [1]

1.6.4 Temporální logika

Zabývá se tvrzeními, ve kterých hraje roli čas, např. "Slunce vychází ráno v osm hodin". [1]

1.6.5 Transparentní intenzionální logika

Pracuje s objekty libovolného řádu, umožňuje rozlišovat tzv. intenze a extenze, přesně explikuje pojem logické konstrukce, definuje, co je to pojem. Umožňuje rozlišovat tři úrovně abstrakce, a to úroveň extenzionální (na které jsou objektem predikace hodnoty funkcí, jakožto zobrazení), intenzionální (kde objektem predikace jsou celé funkce) a hyperintenzionální (kde objektem predikace je příslušná konstrukce funkce). [4]

1.7 Důkazové kalkuly

Logika formalizuje (analyzuje) pojmy jako je tvrzení a úsudek. Formalizace pojmů a práce s nimi je ve skutečnosti závislá na konkrétním logickém kalkulu, se kterým pracujeme. Logický kalkul lze zjednodušeně chápat jako soubor pravidel, která specifikují, jak se formalizace používá. Hlavní cílem takového kalkulu je dokazování logicky pravdivých formulí a platných úsudků. Formule je logicky pravdivá, je-li pravdivá v každé interpretaci, tj. libovolná interpretace je jejím modelem. Úsudek je platný, pokud je závěr úsudku pravdivý ve všech modelech předpokladů. Jinými slovy, předpoklad pravdivosti premis a nepravdivosti závěru vede ke sporu.

Jedna ze základních metod, která rozhoduje o tom, zda daná formule výrokové logiky je tautologie, je takzvaná tabulková metoda. Při tabulkové metodě (tabelaci) vytváříme tabulku, jejíž řádky reprezentují pravdivostní ohodnocení a sloupce reprezentují formule, nebo její podformule. Počet kombinací pravdivostních hodnot, který je nutno vzít v úvahu, roste exponenciálně s počtem atomických výroků. Tuto metodu nelze pro její časovou a paměťovou náročnost efektivně implementovat. [1]

Tautologie lze definovat i syntakticky. Tyto formule lze odvodit mechanickou aplikací jistých strukturálních pravidel. Tautologie jsou přesně ty formule, které lze formálně dokázat pomocí pravidel daného důkazového systému neboli kalkulu. Místo strukturální pravidla budeme říkat odvozovací pravidla.

Axiomy jsou formule, které automaticky přijímáme jako platné. Množinu axiomů a odvozovacích pravidel, která používáme, souhrnně nazýváme *axiomatický systém.*

Odvozovací pravidla našeho kalkulu jsou aplikovatelná na libovolné formule přesného tvaru, bez ohledu na její pravdivostní hodnotu či smysl. Tato pravidla musí být zvolena tak, aby byl kalkul korektní, tedy aby umožňoval odvodit z tautologií opět pouze tautologie, a z daných dodatečných předpokladů pouze to, co z těchto předpokladů logicky vyplývá.

Kalkulus chápeme jako množinu axiomů a odvozovacích pravidel. Kalkulus je korektní, jestliže každá formule v něm dokazatelná je tautologie. Kalkulus je úplný, jestliže je korektní a všechny tautologie jsou v něm dokazatelné. [5]

Obrázek 1: Gerhard Gentzen, zdroj: *Eckart Menzler-Trott, http://owpdb.mfo.de*

1.8 Gerhard Gentzen

V této kapitole čerpám z práce [2], a to zejména v úvodních odstavcích.

Německý logik a matematik Gerhard Karl Erich Gentzen se narodil 24. 11. 1909 v Greifswaldu (Německo - Pomořany). Vyrůstal v Bergenu na Rujáně (Rugen). Jeho otec Hans Gentzen zde působil jako právník, matka Melanie, roz. Bilharozva, pracovala jako učitelka na obchodní škole.

Když bylo Gentzenovi necelých pět let, vypukla 1. světová válka, v necelých deseti letech na jaře v roce 1919 ztratil otce, který zemřel na následky válečného zranění. Matka se v roce 1920 přestěhovala s Gerhardem a jeho sestrou do Stralsundu. Gentzen byl útlé tělesné konstrukce a jeho zdraví bylo od dětství podlomeno. Ve Stralsundu začal studovat na humanistickém gymnáziu. Od mládí se u něho projevovaly silné sklony k matematice, které spolu s výrazným nadáním a velkou pílí z něho učinily jednoho z nejlepších žáků Stralsundského gymnázia.

Gentzen maturoval o velikonocích 1928 a na návrh ředitele ústavu dostal stipendium od tehdejšího Deutsches Stundentenwerk, aby mohl pokračovat ve studiu na univerzitě. Začal studovat matematiku a fyziku, vždy po jednom semestru, na univerzitách v Greifswaldu, Gotinkách (Góttingen), Mnichově, Berlíně a nakonec opět v Gotinkách, kde byl v létě 1933 promován na doktora filozofie.

V roce 1935 se Gerhard Gentzen stal asistentem Davida Hilberta a následně pokračovatelem Hilbertova programu.

V roce 1935 Gerhard Gentzen uvedl kalkul přirozené dedukce. Tehdy popsal všechna pravidla přirozené dedukce pro variantu klasické i intuicionistické predikátové logiky. Přirozená dedukce je také jedním z důkazových kalkulů, ale oproti Hilbertovskému systému se řadí k formálním předpokladovým systémům. V roce 1938 je jmenován mimořádným profesorem na Univerzitě Karlově v Praze.

O Gentzenově práci píše Kriesel cituji: *Jeho pojetí logického důkazu mu umožňuje rozlišovat mezi různými formálními systémy se stejnou množinou konečných teorémů a mezi různými odvozeními v tomtéž systému se stejnou konečnou formulí. (Nejznámější mezi těmito systémy jsou tzv. bezřezové nebo normální systémy.) Jednoduše řečeno, novinkou ve srovnání s analýzou Gottloba Frega je: důkazy reprezentované formálními odvozeními jsou hlavním předmětem studia, ne pouhými nástroji pro analýzu relace důsledku nebo predikátu platnosti. Jinými slovy, studium se týká procesu uvažování, nejen jeho výsledků, tj. dokázaných teorémů.* [6]

Na konci války 5. 5. 1945 je zatčen s ostatními profesory kvůli spojení s NSDAP (Národně socialistická německá dělnická strana). Umírá tři měsíce po zatčení.

Gerhard Gentzen zemřel v r. 1945 ve svých 36 letech. O místě a příčině jeho smrti se historické prameny rozcházejí. Jediné co se shoduje je datum. Například podle životopisu, který sestavila jeho matka, pravděpodobně umírá na vyčerpání v táboře nucených prací. Jiná z verzí je zabití rozvášněným davem Čechů, obviňujícím Genzena z kolaborace s nacisty.

Cituji ze článku Přemysla Vihana: *Jaký nacista? Lépe řečeno, jaký člověk? Byl nacista, ale do války se vůbec nehrnul 19. 11. 1942 byl v Gotinkách pro neschopnost vojenské služby superarbitrován. To v době války znamenalo mnoho a vypovídá to také o jeho vážně porušeném zdraví. Ze slov logiků, s kterými se Gentzen za svého života stýkal, vyplývá, že to byl „neškodný nacista"a podle své povahy hodný člověk. Matematici ho prostě brali, jako by jeho nacismus byl znetvořením, se kterým se nedá nic dělat, asi jako když člověk kulhá.*

Cituji ještě z jeho vojenského spisu z roku 1942: *Použit na domovském bojišti jako radista od 22. 1.1942 do 21.6.1942, potom pobýval v lazaretu a 19. 11. 1942 byl pro neschopnost vojenské služby superarbitován. Politická činost 0, válečné řády 0, čestná vyznamenání 0.*

Z tohoto jasně vyplývá, že politika Gentzena vůbec nezajímala. Jeho hlavním zájmem vždy byla jenom věda. Tehdejší doba byla složitá a záleží vždy na úhlu pohledu, kterým usuzujeme. Pokud se vžijeme na chvíli do role vědce, pro něhož je matematika na samém vrcholu žebříčku hodnot, tak jeho chování se nám zdá v onom kontextu s tehdejší dobou čistě racionální.

Po jeho důkazu bezespornosti čisté teorie čísel, měl následovat důkaz bezespornosti analýzy, tomu ovšem zabránila smrt. Gentzen do poslední chvíle věřil, že se mu to podaří.

2 Gentzenův kalkul

2.1 Popis základních principů Gentzenova kalkulu

Gentzenův kalkul je logický kalkul, který jako svou hlavní myšlenku využívá tento princip: Pokusit sestrojit důkaz dané formule od konce, to znamená, že bychom se pokusili zpětným užíváním pravidel kalkulu dospět od dané formule k axiomům.

V tomto kalkulu nebudeme dokazovat jednotlivé formule, ale sekventy. Proto tento kalkul dostal příznak sekventový. Obecněji si popíšeme základní pojmy používané v tomto kalkulu. Pro tento účel nejprve vše vysvětlím na příkladech výrokové logiky a později v sekci 2.9 na příkladech důkazů formulí predikátové logiky 1. řádu.

2.2 Sekvent

Sekvent je definován jako dvojice konečných množin formulí Γ a Δ. Sekventy zapisujeme ve tvaru $\Gamma \Rightarrow \Delta$. Tento zápis znamená, platí-li všechny formule Γ, pak platí i některé formule v Δ. Symbol dvojité šipky v tomto případě není matematickou značkou, ani logickou spojkou, nýbrž formálním symbolem oddělujícím množiny Γ a Δ. Této dvojité šipce říkáme sekventová šipka. Množinu Γ nazýváme antecedent a množinu Δ sukcedent. Antecedent i sukcedent mohou být i prázdné.

2.3 Příklad zápisu formulí v sekventovém tvaru

Obecný zápis v sekventu:

$$\Gamma \Rightarrow \Delta$$

Neformálně sekvent můžeme chápat jako formule: $A_1, \ldots, A_n$ a $B_1, \ldots, B_m$

$$A_1, \ldots, A_n \Rightarrow B_1, \ldots, B_m$$

Tento zápis koresponduje s:

$$A_1 \wedge ... \wedge A_n \supset B_1 \vee ... \vee B_m$$

Množiny Γ a Δ jsou konečné multisety formulí (daná množina formulí se může vyskytovat opakovaně):

$$A, A, A, A, B \Rightarrow B, A$$

Jak takový sekvent vytvořím? Chceme například v sekventovém kalkulu dokázat tuto formuli :

$$(\mathrm{A} \wedge \mathrm{B}) \supset (\mathrm{B} \vee \mathrm{A})$$

Převedeme ji na sekvent v tomto tvaru:

$$\Rightarrow (A \wedge B) \supset (B \vee A)$$

Takový sekvent je označován jako výsledný, tento sekvent chceme v našem kalkulu dokázat.

2.4 Důkaz v Gentzenově kalkulu

Definujeme ho jako konečnou posloupnost sekventů, kde je z výsledného sekventu každý další odvozen z předchozího pomocí určitého pravidla.

Pro člověka je graficky přehlednější varianta důkazu pomocí konečného orientovaného stromu, který má vrcholy ohodnoceny sekventy. Takovému stromu říkáme sekventový.

2.5 Sekventový strom

Definujeme jako posloupnost sekventů, v níž je každý odvozen z předchozího pomocí některého pravidla. V listech tohoto sekventového stromu jsou iniciální sekventy, neboli axiomy.

Axiomem v tomto případě rozumíme sekvent, který má stejnou formuli současně v antecedentu tak i v sukcedentu např. $(A \Rightarrow A)$ nebo $(B \Rightarrow B)$. Na opačné straně tohoto sekventového stromu je v jeho kořenu výsledný sekvent, v našem příkladě tento $\Rightarrow (A \wedge B) \supset (B \wedge A)$.

Na obrázku uvádím příklad takového sekventového stromu s kořenem dole. Kroky důkazu odpovídající pravidlům jsou vyznačenými vodorovnými linkami, které pro lepší orientaci budeme označovat typem pravidla, které jsme v důkazu použili. Nahoře ve stromu jsou listy s iniciálními sekventy, neboli axiomy.

Příklad 2.1.

$$\dfrac{\dfrac{\dfrac{\dfrac{B \Rightarrow B}{A, B \Rightarrow B}\,LW}{A \wedge B \Rightarrow B}\,L\wedge \qquad \dfrac{\dfrac{A \Rightarrow A}{A, B \Rightarrow A}\,LW}{A \wedge B \Rightarrow A}\,L\wedge}{A \wedge B \Rightarrow B \wedge A}\,R\wedge}{\Rightarrow (A \wedge B) \supset (B \wedge A)}\,R\supset$$

■

V tomto příkladu je finální sekvent odvozen čtyřmi kroky ze dvou axiomů. Nezapomeňme, že důkaz ve skutečnosti sestavujeme pomocí pravidel zespodu vzhůru. Výsledný strom jako logický důkazový kalkul platí potom jen jedním směrem shora dolů, od axiomů k finálnímu segmentu. Obecný tvar jednotlivého kroku důkazu charakterizuje tento předpis:

$$\frac{S_1}{S} \quad \text{nebo} \quad \frac{S_1 \qquad S_2}{S}$$

S_1, S_2 a S jsou sekventy a zároveň S_1, S_2 jsou předpoklady a S je závěr.

Jednotlivé kroky důkazu stanoví logická pravidla sekventového kalkulu. Každé takové logické pravidlo zavádí novou logickou formu, a to pro pravou a levou stranu segmentu v závislosti na pozici sekventové šipky $\Rightarrow$. Proto všechny názvy pravidel začínají písmenem L (left) nebo R (right).

Mějme stále na paměti, že pokud dokazujeme tautologičnost nějakého sekventu, tak i nově utvořené sekventy výše ve stromu jsou také tautologické. Tedy v Gentzenově sekventovém kalkulu postupujeme v důkazu od tautologií k tautologiím.

2.6 Pravidla kalkulu pro výrokovou logiku

Pravidla jsou strukturální nebo logická.

2.6.1 Strukturální pravidla

Tato pravidla pracují na bázi struktury sekventů a nezajímají se o přesný tvar formule. Jedná se o pravidlo oslabení (anglicky weakening označované W).

$$LW\ \frac{\Gamma \Rightarrow \Delta}{A, \Gamma \Rightarrow \Delta} \qquad RW\ \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow A, \Delta}$$

Pravidlo kontrakce (anglicky contraction označované C).

$$LC\ \frac{A, A, \Gamma \Rightarrow \Delta}{A, \Gamma \Rightarrow \Delta} \qquad RC\ \frac{\Gamma \Rightarrow A, A, \Delta}{\Gamma \Rightarrow A, \Delta}$$

Pravidlo řezu (anglicky CUT).

$$CUT\ \frac{\Gamma \Rightarrow \Delta, A \qquad A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}$$

Na pravidlech Gentzenova kalkulu je podstatné, že mohou některé formule postupně zmizet, nebo se některé formule naopak postupně objevit. Formulím, které při použití pravidel vznikají nově, nebo opakovaně se říká *principiální* formule. Formulím, které mizí, říkáme *vstupní* a ostatním formulím beze změn říkáme *postranní* formule. Vše bude lépe vidět na příkladech důkazů v predikátové logice v sekci 2.10.

2.6.2 Logická pravidla

Tato pravidla pracují s logickými spojkami a na rozdíl od strukturálních pravidel berou v potaz přesný tvar formule.

$$L\wedge\ \frac{A, B, \Gamma \Rightarrow \Delta}{A \wedge B, \Gamma \Rightarrow \Delta} \qquad R\wedge\ \frac{\Gamma \Rightarrow \Delta, A \qquad \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \wedge B}$$

$$L\vee\ \frac{A, \Gamma \Rightarrow \Delta \qquad B, \Gamma \Rightarrow \Delta}{A \vee B, \Gamma \Rightarrow \Delta} \qquad R\vee\ \frac{\Gamma \Rightarrow \Delta, A, B}{\Gamma \Rightarrow \Delta, A \vee B}$$

$$L\supset\ \frac{\Gamma \Rightarrow \Delta, A \qquad B, \Gamma \Rightarrow \Delta}{A \supset B, \Gamma \Rightarrow \Delta} \qquad R\supset\ \frac{A, \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \supset B}$$

$$L\neg\ \frac{\Gamma \Rightarrow \Delta, A}{\neg A, \Gamma \Rightarrow \Delta} \qquad R\neg\ \frac{A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg A}$$

2.7 Úplnost kalkulu

Věta o úplnosti Gentzenova kalkulu: *Sekvent* $\Gamma \Rightarrow \Delta$ *je dokazatelný v Gentzenově kalkulu, právě když je tautologický.* [5]

Cituji důkaz implikace $\Rightarrow$ větou o korektnosti, kterou lze dokázat indukcí podle počtu kroků v důkazu sekventu $\Gamma \Rightarrow \Delta$ ([5] s. 44): *Je zřejmé, že každý iniciální sekvent je tautologický. Neexistuje-li pravdivostní ohodnocení v, které přiřazuje hodnotu* 1 *všem formulím v* Γ *a hodnotu* 0 *všem formulím* Δ*, pak neexistuje ani pravdivostní ohodnocení, které přiřazuje hodnotu* 1 *všem formulím v* Γ, Π*, a hodnotu* 0 *všem formulím v* Δ, Λ*, a to bez ohledu na volbu množin formulí* Γ *a* Δ *. Tím je ověřena korektnost pravidla W.*

Korektnost všech ostatních pravidel lze ověřit podobně.

Citace věty o dokazatelnosti tautologických sekventů ([5] s. 44) : *Nechť v množině* $\Gamma \cup \Delta$ *nejsou žádné logické spojky, tzn. jsou tam samé atomické formule. Když* $\Gamma \cap \Delta = \emptyset$*, lze všem formulím v* Γ *přiřadit hodnotu 1 a všem formulím v* Δ *hodnotu 0. Jinak řečeno, když* $\Gamma \Rightarrow \Delta$ *je tautologický sekvent, pak* $\Gamma \cap \Delta \neq \emptyset$ *a sekvent* $\Gamma \Rightarrow \Delta$ *je iniciální, tedy dokazatelný v Gentzenově kalkulu.*

Větu o silné úplnosti Gentzenova kalkulu explicitně neformulujeme, spokojíme se s následujícím komentářem, cituji: ([5] s. 45) *Existují nejméně dva rozumné způsoby, jak Gentzenovský kalkulus definuje dokazatelnost z množiny předpokladů.*

(*i*) *Sekvent je dokazatelný z množiny předpokladů T, jestliže existuje konečná množina* $\Omega \subseteq T$ *taková, že sekvent* $\Omega, \Gamma \Rightarrow \Delta$ *je dokazatelný v Gentzenově kalkulu.*

(*ii*) *Sekvent je dokazatelný z množiny předpokladů T, jestliže je dokazatelný v modifikovaném kalkulu, ve kterém se kromě běžných iniciálních sekventů připouštějí ještě iniciální sekventy tvatu* $\Rightarrow \varphi$*, kde* $\varphi \in T$*.*

2.8 Příklady důkazů výrokové logiky

Zde uvedu pár možností jak daný důkaz sestrojit. Vezmeme si prozatím formuli výrokové logiky $(A \wedge B) \wedge A \supset (B \vee A) \vee B$. Pomocí pravidel Gentzenova kalkulu sestrojme důkazy tautologičnosti této formule. Pro názornost jsem použil několik různých řešení. Všechny tyto stromy jsou platné důkazy a jsou navzájem ekvivalentní. Liší se jen typem a pořadím použitých pravidel.

Příklad 2.2. *Důkaz formule s pravidlem CUT:*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A \Rightarrow A}{A \Rightarrow B, A, C}\,LW}{A \Rightarrow B \vee A, C}\,R\vee \qquad \dfrac{\dfrac{B \Rightarrow B}{C, A, B \Rightarrow B}\,RW}{C, A \wedge B \Rightarrow, B}\,L\wedge}{A, A \wedge B \Rightarrow B \vee A, B}\,CUT}{A \wedge (A \wedge B) \Rightarrow B \vee A, B}\,L\wedge}{A \wedge (A \wedge B) \Rightarrow (B \vee A) \vee B}\,R\vee}{\Rightarrow A \wedge (A \wedge B) \supset (B \vee A) \vee B}\,L\supset$$

■

Důkaz formule s pravidlem Contraction:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{B, A, A, B \Rightarrow B, A, A, B}{B, A, A, B \Rightarrow B, A, B}\,RC}{A, A, B \Rightarrow B, A, B}\,LC}{A, A, B \Rightarrow B \vee A, B}\,R\vee}{A, (A \wedge B) \Rightarrow B \vee A, B}\,L\wedge}{A \wedge (A \wedge B) \Rightarrow B \vee A, B}\,L\wedge}{A \wedge (A \wedge B) \Rightarrow (B \vee A) \vee B}\,R\vee}{\Rightarrow A \wedge (A \wedge B) \supset (B \vee A) \vee B}\,L\supset$$

■

Důkaz formule s pravidlem Weakening:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A, B \Rightarrow A, B}{RW}}{A, B \Rightarrow B, A, B}}{LW}}{A, A, B \Rightarrow B, A, B}}{R\vee}}{A, A, B \Rightarrow B \vee A, B}}{L\wedge}}{A, (A \wedge B) \Rightarrow B \vee A, B}}{L\wedge}}{A \wedge (A \wedge B) \Rightarrow B \vee A, B}}{R\vee}}{A \wedge (A \wedge B) \Rightarrow (B \vee A) \vee B}}{L \supset}}{\Rightarrow A \wedge (A \wedge B) \supset (B \vee A) \vee B}$$

■

Vraťme se ještě k citaci důkazu o větě o úplnosti kalkulu. Důkaz daného sekventu se sestrojuje zpětným používáním pravidel. Na příkladech je názorně vidět, že ne všechna pravidla jsme v důkazu potřebovali. Ve třetím stromu jsme se obešli bez pravidel C a CUT a věta o úplnosti by tedy platila i pro kalkul bez těchto dvou pravidel.

Neznamená to ovšem, že by se zde tato pravidla vyskytovala zbytečně. Naopak, například pravidlo CUT může některé důkazy velmi rychle zkrátit. Vše závisí na zkušenosti člověka (řekněme s trochou nadsázky, jakou daný člověk použije strategii, jakou důkaz povede) a správném pořadí použití jednotlivých pravidel. Pravidlo CUT je velice potřebné, zejména pro formule predikátové logiky 1. řádu.

Záměrně jsme v příkladu použili tuto triviální formuli, abychom na ní ukázali všechny možné varianty sestavování důkazu. Pro názornou představu o složitosti sestrojení univerzálního algoritmu, který by tento kalkul řešil, by tato ukázka měla stačit. Později tento algoritmus i s optimalizací vysvětlím v sekci 3.

2.9 Pravidla kalkulu pro predikátovou logiku

Pro tuto logiku platí všechna pravidla popsaná výše. Nyní množiny Γ, Δ jsou množiny predikátových formulí. Dále budeme používat čtyři kvantifikátorová pravidla.

$$L\forall \frac{A[x/t], \Gamma \Rightarrow \Delta}{\forall x A(x), \Gamma \Rightarrow \Delta} \qquad R\forall \frac{\Gamma \Rightarrow A[x/y], \Delta}{\Gamma \Rightarrow \forall x A(x), \Delta} *gen$$

$$L\exists \frac{A[x/y], \Gamma \Rightarrow \Delta}{\exists x A(x), \Gamma \Rightarrow \Delta} *gen \qquad R\exists \frac{\Gamma \Rightarrow A[x/t], \Delta}{\Gamma \Rightarrow \exists x A(x), \Delta}$$

V případě pravidel označené $*gen$ to jest $L\exists$ a $R\forall$ je proměnná y substituovatelná za x ve formuli A a nemá žádné volné výskyty v množině Γ a Δ. U pravidel $L\forall$ a $R\exists$ je term t substituovatelný za x.

Všimněte si, že u všech čtyř kvantifikovaných pravidel máme co dělat s dosazením za proměnou a že dosazení vždy směřuje proti směru úvahy. Abychom ověřili, že formule $\exists x A(x)$ nebo $\forall x A(x)$ je správně odvozená ze vstupní formule, musíme ověřit, že tuto vstupní formuli můžeme získat z formule A (tj. z té formule, kterou získáme z principiální formule odstraněním kvantifikátoru prvního v řadě), dosadíme za x (tj. za tu proměnou, která je určena oním kvantifikátorem). [5]

Pravidlům $L\exists$ a $R\forall$ říkáme pravidla generalizace, pravidlům $R\exists$ a $L\forall$ pravidla konkretizace.

Zde uvedeme jeden neformální důkaz cituji Vítězslava Švejdara: ([5] s. 183)

Pravidlo $L\exists$ je formalizací následujícího kroku:

. . . Máme zdůvodnit, že platí Δ, přičemž víme, že existuje objekt s vlastností φ. Zvolme takový objekt a označme jej y. Stačí zdůvodnit, že Δ platí za předpokladu $\varphi x(y)$.

Analogicky je pravidlo $R\forall$ formalizací takovéhoto kroku:

. . . Máme zdůvodnit, že všechny objekty mají vlastnost φ. Nechť je tedy dán nějaký objekt, označme jej y. Stačí zdůvodnit $\varphi x(y)$.

Oba kroky jsou správné za předpokladu, že y zatím nic neoznačuje. Tomu odpovídá podmínka u pravidel generalizace, že y se nevyskytuje volně v množinách Γ a Δ ani ve formuli $\exists x\varphi$ resp.$\forall x\varphi$. Tato podmínka bývá v literatuře označena německo-anglickým názvem eigenvariable condition. V našem textu jí říkejme podmínka EVC. Všimněme si ještě, že pravidla generalizace

připouštějí, aby x a y byla tatáž proměnná. V tom případě se pravidla $L\exists$ a $R\forall$ podobají pravidlům Gen-E a Gen-A Hilbertovského kalkulu a je automaticky splněno, že proměnná y nemá volné výskyty ve formuli $\exists x\varphi$ resp.$\forall x\varphi$.

Nejlépe vše vysvětlíme na konkrétních příkladech. Pro lepší pochopení zde uvádíme jen sekventy bez logických spojek. Nezapomeňte, že důkaz sestavujeme odspodu. Všimněte si, že nejprve odvodíme z formule $\exists x P(x)$ formuli $P(x)$ použitím pravidla $R\exists$. Až poté můžeme použít pravidlo $L\forall$, v opačném pořadí by to nebylo možné, nebyla by splněna podmínka EVC.

Příklad 2.3. *obsahující všeobecné i existenční kvantifikátory:*

$$\dfrac{\dfrac{P(a) \Rightarrow P(a)}{\forall x P(x) \Rightarrow P(a)}\, L\forall}{\forall x P(x) \Rightarrow \exists y P(y)}\, R\exists$$

■

Příklad 2.4. *obsahující jenom všeobecné kvantifikátory:*

$$\dfrac{\dfrac{P(x) \Rightarrow P(x)}{\forall x P(x) \Rightarrow P(x)}\, R\forall}{\forall x P(x) \Rightarrow \forall y P(y)}\, R\forall$$

■

Příklad 2.5. *obsahující jenom existenční kvantifikátory:*

$$\dfrac{\dfrac{P(y) \Rightarrow P(y)}{P(y) \Rightarrow \exists y P(y)}\, P\exists}{\exists x P(x) \Rightarrow \exists y P(y)}\, L\exists$$

■

Uvedeme ještě jednu možnou kombinaci kvantifikátorů. Níže popsaný strom důkazem ovšem není. Zde byla podmínka EVC porušena. Formule $\exists x P(x) \supset \forall y P(y)$ není logicky platná. Proto nemá sekvent $\exists x P(x) \Rightarrow \forall y P(y)$v Gentzenově kalkulu žádný důkaz. Tuto formuli můžeme interpretovat: "některá P jsou všechna P", což je očividně špatně.

Příklad 2.6.

$$\dfrac{\dfrac{P(x) \Rightarrow P(x)}{\exists x P(x) \Rightarrow P(x)}\, R\exists}{\exists x P(x) \Rightarrow \forall y P(y)}\, R\forall$$

Celý tento proces eliminace kvantifikátorů Gentzenova kalkulu v sobě absorbují principy unifikace. Správné používání pravidel pro kvantifikátory vyžaduje zvláštní pečlivost. Celý postup vysvětlím v sekci 3.5 na příkladech pro eliminaci kvantifikátorů. Na tomto místě zmíníme jen lemma o substituci, cituji Vítězslava Švejdara ([5] s. 192):

Nechť P je důkaz, nechť z je proměnná, která v důkazu P není generalizována, nechť s je term, jehož žádná proměnná není v důkazu P generalizována ani kvantifikována. Pak $Pz(s)$, výsledek substituce termu s za všechny volné výskyty proměnné z v důkazu P, je opět důkazem.

Cituji myšlenku substituce: *Není-li žádná proměnná termu s v důkazu P kvantifikována, pak term s je v každé formuli důkazu P substituovatelný za z. Pravidla generalizace umožňují generalizovat proměnné, nikoliv termy. Na tom se ale nic nepokazí, neboť term s nedosazujeme za proměnnou, která je kdekoliv v důkazu P generalizována. Protože žádná proměnná termu s není v důkazu P generalizována, substituce termu s nezanese nežádoucí volné proměnné do žádného místa, kde je v P použito pravidlo generalizace, tj. nikde nepokazí platnost podmínky EVC. QED.*

Obecný princip užívání kvantifikátorových pravidel vyjadřuje tento strom:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\phi \Rightarrow \phi}{\forall v \phi \Rightarrow \phi}\, L\forall}{\forall v \phi \Rightarrow \exists u \phi}\, R\exists}{\exists u \forall v \phi \Rightarrow \exists u \phi}\, L\exists\ (u \text{ je vázaná napravo})}{\exists u \forall v \phi \Rightarrow \forall v \exists u \phi}\, R\forall\ (v \text{ je vázaná nalevo})}{\Rightarrow \exists u \forall v \phi \supset \forall v \exists u \phi}\, R\supset$$

2.10 Příklady důkazů predikátové logiky

Nyní můžeme uvést bez pochyb trošku složitější důkaz, například formule $\exists x(A(x) \supset B) \supset \forall y A(y) \supset B$ má v Gentzenově kalkulu takový sekventový strom. Strom končí axiomem. Formule je tedy dokazatelná.

Příklad 2.7.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A(y) \Rightarrow A(y)}{\forall y A(y) \Rightarrow A(y)}\, L\forall \qquad B \Rightarrow B}{A(y) \supset B, \forall y A(y) \Rightarrow B}\, L\supset}{\exists x(A(x) \supset B), \forall y A(y) \Rightarrow B}\, L\exists}{\exists x(A(x) \supset B) \Rightarrow \forall y A(y) \supset B}\, R\supset}{\Rightarrow \exists x(A(x) \supset B) \supset \forall y A(y) \supset B}\, R\supset$$

■

Například formule $\exists x A(x) \vee \exists x B(x) \supset \exists y((A)(y) \vee B(y))$ je v Gentzenově kalkulu dokazatelná. Má takový sekventový strom.

Příklad 2.8.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A(y) \Rightarrow A(y)}{A(y) \Rightarrow A(y), B(y)}\, RW}{A(y) \Rightarrow A(y) \vee B(y)}\, R\vee}{A(y) \Rightarrow \exists y(A(y) \vee B(y))}\, R\exists}{\exists x A(x) \Rightarrow \exists y(A(y) \vee B(y))}\, L\exists \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{B(y) \Rightarrow B(y)}{B(y) \Rightarrow A(y), B(y)}\, RW}{B(y) \Rightarrow A(y) \vee B(y)}\, R\vee}{B(y) \Rightarrow \exists y(A(y) \vee B(y))}\, R\exists}{\exists x B(x) \Rightarrow \exists y(A(y) \vee B(y))}\, L\exists}{\exists x A(x) \vee \exists x B(x) \Rightarrow \exists y((A)(y) \vee B(y))}\, L\vee}{\Rightarrow \exists x A(x) \vee \exists x B(x) \supset \exists y((A)(y) \vee B(y))}\, R\supset$$

■

Jak jsme již zmínili, existuje mnoho variant, jak jednotlivý důkaz provedeme. Pokud jsou pravidla užitá korektně, jsou důkazy ekvivalentní. Například formuli $(\forall x A(x) \supset B) \supset \exists y(A(y) \supset B)$ můžeme dokázat jednak s pravidlem C ale i bez něj.

Příklad 2.9. *důkaz s pravidly Contraction:*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A(y) \Rightarrow A(y)}{A(y) \Rightarrow A(y), B}\,RW}{\Rightarrow A(y), A(y) \supset B}\,R\supset}{\Rightarrow A(y), \exists y(A(y) \supset B)}\,R\exists}{\Rightarrow \forall x A(x), \exists y(A(y) \supset B)}\,R\forall \qquad B \Rightarrow B}{\forall x A(x) \supset B \Rightarrow \exists y(A(y) \supset B), B}\,L\supset}{\forall x A(x) \supset B, A(a) \Rightarrow \exists y(A(y) \supset B), B}\,LW}{\forall x A(x) \supset B \Rightarrow \exists y(A(y) \supset B), A(a) \supset B}\,R\supset}{\forall x A(x) \supset B \Rightarrow \exists y(A(y) \supset B), \exists x(A(x) \supset B)}\,R\exists}{\forall x A(x) \supset B \Rightarrow \exists y(A(y) \supset B)}\,RC}{\Rightarrow (\forall x A(x) \supset B) \supset \exists y(A(y) \supset B)}\,R\supset$$

■

Důkaz bez pravidel Contraction:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A(y) \Rightarrow A(y)}{A(y) \Rightarrow A(y), B}\,RW}{\Rightarrow A(y), A(y) \supset B}\,R\supset}{\Rightarrow A(y), \exists y(A(y) \supset B)}\,R\exists}{\Rightarrow \forall x A(x), \exists y(A(y) \supset B)}\,R\forall \qquad \dfrac{\dfrac{\dfrac{B \Rightarrow B}{A(a), B \Rightarrow B}\,LW}{B \Rightarrow A(a) \supset B}\,R\supset}{B \Rightarrow \exists x(A(y) \supset B)}\,R\exists}{\forall x A(x) \supset B \Rightarrow \exists y(A(y) \supset B)}\,L\supset}{\Rightarrow (\forall x A(x) \supset B) \supset \exists y(A(y) \supset B)}\,R\supset$$

■

Nyní uvedeme důkaz pro formuli $\neg\exists x A(x) \supset \forall y \neg A(y)$ s využitím pravidla CUT. Tato formule je v Gentzenově kalkulu dokazatelná. Každá taková formule dokazatelná prostřednictvím pravidla CUT má i bezřezový důkaz. Tuto formuli proto musíme dokázat i bezřezově.

Příklad 2.10. *s pravidlem CUT:*

$$\dfrac{\dfrac{\dfrac{A(x) \Rightarrow A(x)}{A(x) \Rightarrow \exists x A(x)}\,R\exists \qquad \dfrac{\exists x A(x) \Rightarrow \exists x A(x)}{\exists x A(x), \neg\exists x A(x) \Rightarrow}\,L\neg}{\dfrac{\dfrac{\dfrac{A(x), \neg\exists x A(x) \Rightarrow}{\neg\exists x A(x) \Rightarrow \neg A(x)}\,R\neg}{\neg\exists x A(x) \Rightarrow \forall y \neg A(y)}\,R\forall}{\Rightarrow \neg\exists x A(x) \supset \forall y \neg A(y)}\,R\supset}\,CUT$$

■

Důkaz bez pravidla CUT:

$$\dfrac{\dfrac{\dfrac{\dfrac{A(x) \Rightarrow A(x)}{A(x) \Rightarrow \exists x A(x)}\,R\exists}{\neg\exists x A(x), A(x) \Rightarrow}\,L\neg}{\neg\exists x A(x) \Rightarrow \neg A(x)}\,R\neg}{\neg\exists x A(x) \Rightarrow \forall y \neg A(y)}\,R\forall \qquad \dfrac{}{\Rightarrow \neg\exists x A(x) \supset \forall y \neg A(y)}\,R\supset$$

■

Existuje věta o eliminovatelnosti řezu, cituji ([5] s. 46): *Každý sekvent dokazatelný v Gentzenově kalkulu je dokazatelný i bez užití pravidla řezu.* Důkaz této věty je poměrně zdlouhavý a pro potřeby hledání jednoduchého algoritmu ke Gentzenově kalkulu je pro naše potřeby nepodstatný. Jeho přesné znění najdete v Keleeneho knize.[7] Nás bude zajímat jen fakt, že toto pravidlo můžeme pro potřeby algoritmizace kalkulu vyloučit, aniž bychom kalkul nějak oslabili, ale o tom více v sekci 3.3: Redukce pravidel.

2.11 Negace

V Gentzenově kalkulu pravidlo pro $L\neg, R\neg$ aplikujeme tak, že jen převedeme přesný tvar formule na druhou stranu sekventové šipky. Přesně jako je popsáno v příkladu 2.10. Nepoužíváme funkci negace jako takovou, ve které při negování zaměňujeme kvantifikátory, nebo měníme konjunkci na disjunkci atd.

V Gentzenově kalkulu totiž vycházíme z důkazů kontrapozice. Nezapomeňme, že vlastní sekvent je chápán neformálně takto ($A \vdash B$). Tyto důkazy kontrapozice vypadají tak, že pokud například chceme dokázat, že za předpokladu B platí A, tak předpokládáme, že neplatí A (tj. že platí ¬A), a dokážeme, že pak neplatí B (tj. platí ¬B). Různé varianty tohoto postupu jsou znázorněny následujícími čtyřmi pravidly.

$$\frac{\Gamma, A \vdash B}{\Gamma, \neg B \vdash \neg A}, \quad \frac{\Gamma, A \vdash \neg B}{\Gamma, B \vdash \neg A}, \quad \frac{\Gamma, \neg A \vdash B}{\Gamma, \neg B \vdash A}, \quad \frac{\Gamma, \neg A \vdash \neg B}{\Gamma, B \vdash A}$$

3 Algoritmus pro Gentzenův kalkul

Výše v textu jsem již zmínil, jak takový kalkul funguje a nyní se budu snažit stručně popsat algoritmus pro automatické dokazování v tomto kalkulu, který jsem sám navrhl a implementoval do softwarového nástroje.

3.1 Syntaxe a sémantika

Pro formule musí být předem dána nějaká konkrétní přesná pravidla, podle kterých je formule možné vytvářet, a která určují, co je a co není dobře vytvořená formule. Algoritmus by vlastně nepoznal, že vstupní text je logická formule, natož aby s ním korektně pracoval.

Pravidla, která určují, jak formule vypadají se nazývají syntaxe. Oproti tomu sémantika přiřazuje těmto formulím nějaký význam, určuje co znamenají jednotlivé symboly, jakých pravdivostních hodnot mohou nabývat.

3.2 Syntaktický analyzátor

Jednou z klíčových věcí pro algoritmizaci logického důkazového kalkulu je syntaktický analyzátor, anglicky takzvaný parser.

Při syntaktické analýze se vstupní text transformuje na určité datové struktury, většinou syntaktický nebo derivační strom, které zachovávají hierarchické uspořádání vstupních symbolů, které jsou vhodné pro další zpracování. [8]

Syntaktické analýze předchází lexikální analýza, při níž se vstupní text rozděluje na posloupnost lexikálních symbolů neboli tokenů. V našem případě literály jazyka predikátové logiky. Pro parser to jsou dále nedělitelné stavební jednotky, které používá při interpretaci vstupních dat. V praxi parsery nebývají programovány ručně, ale pomocí zvláštních programů tzv. parser generátorů, pomocí bezkontextových gramatik.

Pro náš účel, parsování textu obsahujícího logickou formuli, se nám jeví výhodnější si takový parser naimplementovat sami. Potřebujeme si totiž vytvořit vlastní objekty, na které budeme moci jednoduše aplikovat pravidla tohoto kalkulu. Lépe se potom budeme orientovat ve složitějších datových strukturách a mnohem přehlednější a lepší bude pro programátora pozdější validace takové datové struktury, která nám slouží jako univerzální nosič našeho vstupního textového řetězce s logickou formulí. Nyní se náš algoritmus pokusím stručně popsat.

Výsledný strom je reprezentovaný datovou strukturou (tzv. kolekcí), která obsahuje jeden typ objektů, pracovně ho nazvěme segment (část). Tento datový objekt má několik slotů nesoucích informace o povaze a určení typu našeho objektu. Jedním z takových slotů je i další kolekce

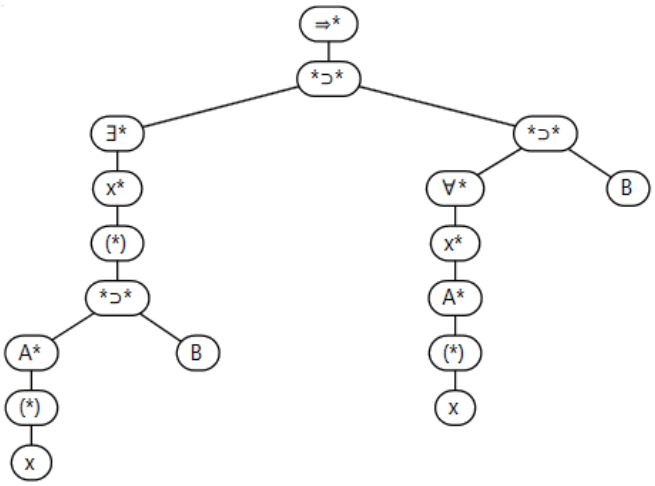
textových symbolů, reprezentující vstupní textový řetězec. O jeho dělení na další objekty se starají dva typy rekurzivních metod.

První metoda nám rozdělí náš segment na tři, neboli vytvoří další dvě větve v našem derivačním stromu. Stará se o dělení logické formule podle duálních logických funkcí ($\supset, \vee, \wedge$), ale i rozdělení podle sekventové šipky, nebo oddělovací čárky v antecedentu nebo sukcedentu.

Jiná metoda nám rozdělí náš segment v určitém bodě textu na dalšího následníka, neboli vytvoří jen jednu větev v derivačním stromu. Stará se o rozdělení podle unárních logických funkcí ($\neg$) nebo kvantifikátorů, ale i predikátových symbolů jazyka, rozdělení pomocí závorek, nebo symbolů zastupujících proměnné v jazyce predikátové logiky 1. řádu.

Jednotlivé segmenty v sobě nesou informaci prostřednictvím číselného ukazatele, který udává identifikátor následujícího segmentu, nebo identifikátor rodičovského segmentu. Takto vytvořená datová struktura po grafické vizualizaci nám znázorní derivační strom textového řetězce. Jestli se bude jednat o správně utvořenou formuli o to se stará jiná metoda, která již pracuje s tímto derivačním stromem. Za pomocí ukazatelů se mohou pomocné metody po tomto stromě pohybovat v jakémkoli směru a hledat, nebo různě upravovat jednotlivé uzly.

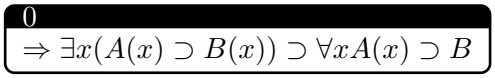
Příklad převodu takového textu s dobře utvořenou logickou formulí v sekventovém tvaru vidíte na obrázku níže. Na grafické reprezentaci derivačního stromu jsou ukazatele znázorněny hvězdičkou. Sekvent ve tvaru $\Rightarrow \exists x(A(x) \supset B) \supset \forall x A(x) \supset B$ má tento derivační strom.



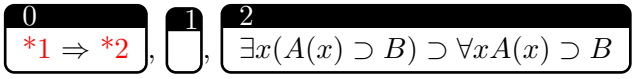
Jednotlivé uzly takto vytvořeného derivačního stromu se zkontrolují pomocí regulárních výrazů. Určí se možné chybné symboly, párování závorek, proměnné u kvantifikátorů, atd. V tomto konkrétním příkladě je syntaxe v pořádku. Celý tento proces si ale lépe představíme v následující ukázce, kde je graficky znázorněna kolekce objektů. Objekty jednotlivé (segmenty) znázorním černým obdélníkem a číslo v hlavičce je jeho identifikátor, který nám bude sloužit jako ukazatel. V obdélníku je textový řetězec. Černý text je v procesu chápán jako literát daného

jazyka a červený text je ukazatel na další segment. Proces tvorby derivačního stromu je rozdělen do osmi kroků, jejich pořadí je navzájem nezaměnitelné.

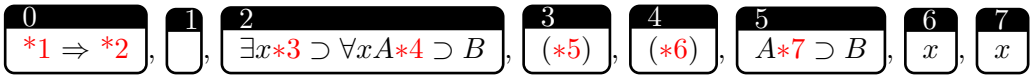
1. Inicializace objektu s textem



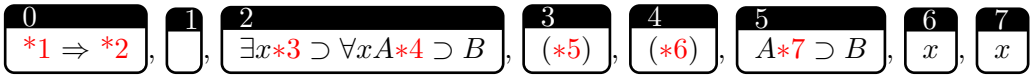
2. Rozdělení podle sekventové šipky



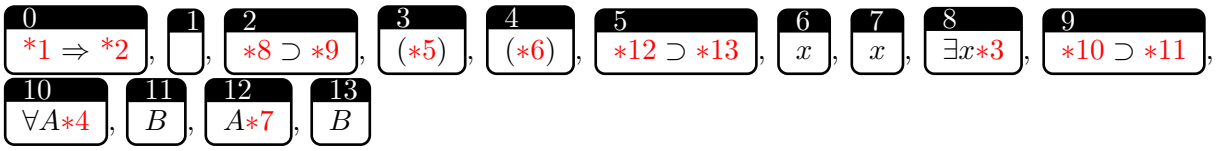
3. Rozdělení podle závorek, bere v úvahu jen typ otevírací nebo uzavírací závorka.



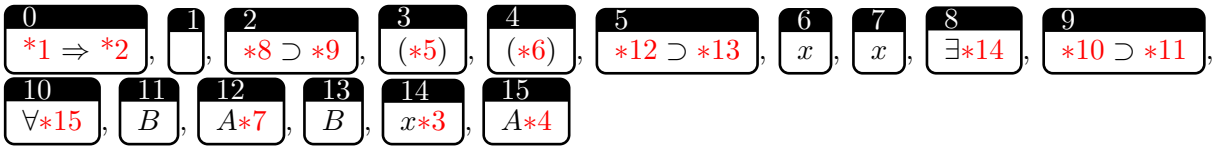
4. Rozdělení podle čárek. V našem příkladu bude kolekce beze změny.



5. Rozdělení podle binárních logických operací. Pokud je v jednom segmentu více logických spojek, postupuje se podle nejmenší priority $(\supset, \vee, \wedge)$.



6. Rozdělení podle kvantifikátorů.



7. Rozdělení podle logické spojky negace. V našem příkladu bude kolekce beze změny.

8. V posledním kroku algoritmu proběhne analýza jednotlivých segmentů, pomocí regulárních výrazů. Na tomto místě se vyhodnotí syntaktické chyby a doplní, popřípadě uberou vnější závorky. Doplní se sloty k jednotlivým objektům, které nám určují literáty jazyka, nebo různé druhy přívlastků v derivačním stromu, například arita predikátových symbolů atd.

Takto interpretovaná struktura derivačního stromu nám vždy zajistí levý podstrom na indexu 1, pravý podstrom na indexu 2.

Je třeba ještě ošetřit některé vstupy. Například pokud zadaný textový řetězec obsahuje dvě sekventové šipky, to znamená, že jsou dva sekventy na jednom řádku. V takovém případě algoritmus musí bezpečně poznat, na jakém místě v textu končí levý sekvent a kde začíná pravý sekvent. To se provede tak, že se ve stromu projedou všechny uzly nalevo od druhé sekventové šipky a tam, kde regulární výraz zjistí v textu některého objektu mezeru, bere tento bod jako dělící prvek obou stromů.

Parser, implementovaný pro tento kalkul, rozpoznává symboly jazyka výrokové nebo predikátové logiky 1. řádu. Jejich přesný výčet je:

1. Individuové proměnné - značí se symboly x, y, z případně s indexy x1, z100 ...
2. Individuové konstanty - značíme malými písmeny (a, b, c, ...) případně s indexy.
3. Funkční symboly - ty budeme označovat malými písmeny f(), g(), h(), ..., stačí pokud za malým písmenem bude závorka.
4. Predikátové symboly - začínají velkými písmeny P, Q, Love, ...
5. Logické spojky - $\neg, \wedge, \vee, \supset, \equiv$
6. Kvantifikátory - $\forall, \exists$
7. Závorky - $(,), [,], \{,\}$, které jsou použity jako pomocné symboly pro explicitní vyjádření priorit, případně pro lepší čitelnost výrazů.

3.3 Redukce pravidel

Pro automatizované dokazování v Gentzenově kalkulu je potřeba zredukovat některá pravidla pro jejich časovou složitost nebo náročnou algoritmizovatelnost. Veškeré redukce musí být ekvivalentní a nesmí narušit korektnost a úplnost Gentzenova kalkulu. Podívejme se nejprve na strukturální pravidla. Například pravidlo CUT můžeme bez obav vyloučit, což jsme si dokázali pomocí věty o eliminovatelnosti řezu. Pravidlo CUT je poměrně náročně algoritmizovatelné a proto ho tedy vyloučíme. Tím se nám v některých případech sekventový strom prodlouží, ale algoritmicky se tento proces výrazně zjednoduší. Další strukturální pravidlo Contraction (označované C) by mělo za následek tvorbu nekonečných výpočetních větví daného důkazu, pro levou nebo pravou stranu sekventu, proto nám vypuštění tohoto pravidla celý proces výrazně výpočetně zjednoduší. Jediné strukturální pravidlo W ponecháme, ale jen s podmínkou použití v konečných listech stromu kvůli lepší čitelnosti axiomu. Jinak by mělo jeho bezhlavé využívání,

někde hluboko vně sekventového stromu, fatální následky. Některé formule by se nám mohly začít ztrácet a režie algoritmu, která by tento proces hlídala, by byla zase časově náročná.

Ve skutečnosti jsem již na začátku takovou malou redukci pravidel provedl, v originálním znění Gentzenova kalkulu pravidla $L\wedge$ a $R\vee$ mají tento předpis:

$$L\wedge \frac{A_i, \Gamma \Rightarrow \Delta}{A_0 \wedge A_1, \Gamma \Rightarrow \Delta}\ (i = 0, 1\ ) \qquad R\vee \frac{\Gamma \Rightarrow \Delta, A_i}{\Gamma \Rightarrow \Delta, A_0 \vee A_1}\ (i = 0, 1\ )$$

Takto definovaná pravidla nám část formule A odmažou, proto se musí kombinovat s pravidlem Contraction. Zvolil jsem jejich jednodušší verzi.

$$L\wedge \frac{A, B, \Gamma \Rightarrow \Delta}{A \wedge B, \Gamma \Rightarrow \Delta} \qquad R\vee \frac{\Gamma \Rightarrow \Delta, A, B}{\Gamma \Rightarrow \Delta, A \vee B}$$

Další redukce jiných pravidel bych již nedoporučoval. Chceme totiž algoritmizovat celý Gentzenův kalkul, tak jak ho navrhl. Další zavádění redukcí by nám tento kalkul spíše směřoval do jiných logických kalkulů nebo algoritmů, například Skolemizace nebo unifikace. Proto si vystačíme jen s redukcí strukturálních pravidel, která nemají na korektnost a úplnost výsledného kalkulu žádný vliv.

3.4 Hlavní rysy algoritmu

Hlavní princip pro správné a korektní aplikování pravidel je jejich použití na správném místě a ve správný čas. Proto je pro každou formuli na vrcholu sekventového stromu potřeba sestavit derivační strom. V kořenu tohoto derivačního stromu je buď logická spojka nebo kvantifikátor. Pro tyto symboly se pak může použít příslušné pravidlo v závislosti na postavení sekventové šipky (pravidla pro levou nebo pravou stranu).

Samostatný sekvent obsahuje sice minimálně jednu formuli, ale obecně jich může být libovolné spočetné množství. V takovou chvíli máme na jedné úrovni sekventového stromu několik derivačních podstromů a několik možností využití konkrétních pravidel. Jak poznat, které pravidlo aplikovat první, aby algoritmus pro automatické dokazování vedl ke zdárnému cíli? Problém jsem vyřešil zavedením prioritního systému. Logické spojky mají přednost před kvantifikátory, protože výsledné sekventy po eliminaci spojek v jakémkoliv pořadí jsou vždy navzájem ekvivalentní. Nicméně pro lepší čitelnost sekventového stromu mají i pravidla pro logické spojky

toto prioritní pořadí: $L \supset, R \supset, L\vee, R\vee, L\wedge, R \wedge L\neg, R\neg$. Pořadí je odvozeno obecně z priorit pro spojky, jenom obráceně. Důkaz sestavujeme od spodu nahoru. Po eliminaci logických spojek přijdou na řadu kvantifikátory, u těch je to mnohem složitější.

3.5 Příklady eliminace kvantifikátorů

Kvůli platnosti podmínky EVC je pořadí jejich eliminace následovné: $R\forall, L\exists, R\exists, L\forall$. Nejprve je nutné použít pravidla generalizace a poté konkretizace. Nutno ještě pohlídat na jaké proměnné se ony kvantifikátory vážou a jestli tyto proměnné mají výskyty i v jiných formulích onoho sekventu. Stručně řečeno, pokud je to možné, nejprve uplatníme pravidlo pro generalizaci $R\forall$ nebo $L\exists$. Popis mojí implementace se vám pokusím stručně popsat na konkrétním příkladu.

Uvedu příklad formule $\exists x_1 \forall y_1 P(x_1, y_1) \supset \forall y_2 \exists x_2 P(x_2, y_2)$, převedu ji rovnou na segment bez logických spojek a přesně popíšu můj algoritmus. V popisu algoritmu se budu snažit co nejméně používat terminologii matematické logiky (jako je výraz term nebo proměnná), ale budu užívat jen termín symbol, proto abych přesně simuloval vlastnosti onoho procesu v paměti počítače. Algoritmus syntaktického analyzátoru (výše popsaný) nám generuje jeden derivační strom. Objekty, které ho reprezentují, mají přiděleny příznak levá nebo pravá část. Pro lepší srozumitelnost si ho fiktivně rozdělím na levý derivační strom a pravý derivační strom podle sekventové šipky. Máme tedy zadaný takový sekvent reprezentovaný textovým řetězcem:

$$\exists x_1 \forall y_1 P(x_1, y_1) \Rightarrow \forall y_2 \exists x_2 P(x_2, y_2)$$

V levé formuli je v kořenu jejího derivačního stromu $\exists$ a v pravé formuli $\forall$. Podle priority má přednost pravý $\forall$. Proto jako první uplatníme pravidlo $R\forall$. To je pravidlo generalizace. Implementovaná funkce generalizace nejprve zjistí, který symbol je přítomný u kvantifikátoru, který chceme eliminovat. Zjistí jeho vazbu na konkrétní predikátové symboly a ty si uloží do pomocné metody. Ukládá si i jejich aritu a index (místo výskytu), kde se symbol proměnné v predikátu nalézá. V tomto příkladě je v pomocné metodě uložen predikát: P, arita: 2, symbol: y_2, index: 2.

Dalším krokem implementované funkce generalizace je zjistit, jestli někde v textovém řetězci je stejný predikátový symbol se stejnou aritou. Pokud ano, podívá se místo kam ukazuje jeho index pro výskyt hledaného symbolu. Tento symbol si zapamatuje a podívá se, jestli je kvantifikován. Pokud ano, jedná se jistě o symbol označující proměnnou, proto může moje metoda tento symbol převzít a použít pro eliminaci původního kvantifikátoru. Tak symbol označující původní proměnnou přepíše symbolem převzatým. Tímto procesem se zajistí substituce proměnné za

proměnnou. Podmínka EVC nebude porušena a takový krok nám v mém algoritmu zaručí korektnost. V našem příkladě to bude symbol y_1 kvantifikovaný nalevo všeobecným kvantifikátorem. Pokud se takový kvantifikovaný symbol nenajde, je nutné postupovat jinak, ale to si vysvětlíme na dalším příkladu na následující straně. V našem příkladu výsledek bude tento.

$$\exists x_1 \forall y_1 P(x_1, y_1) \Rightarrow \exists x_2 P(x_2, y_1)$$

V dalším kroku je v novém derivačním stromu nalevo kořen $\exists$ a napravo $\exists$. Podle priority má přednost levý $\exists$. Uplatníme pravidlo $L\exists$. To je pravidlo generalizace, které postupuje tak, jak jsme si vysvětlili výše. V našem příkladu výsledný řetězec bude tento.

$$\forall y_1 P(x_2, y_1) \Rightarrow \exists x_2 P(x_2, y_1)$$

Dále je v kořenech nalevo $\forall$ a napravo $\exists$. Prioritu má pravý $\exists$. Uplatníme pravidlo $R\exists$, to je konkretizace. Zjistí pomocí derivačního stromu, který symbol se váže na eliminovaný kvantifikátor a jeho vazbu na konkrétní predikátové symboly. Do pomocné metody si uloží ony predikátové symboly, jejich aritu a index. V tomto příkladu je v pomocné metodě uložen predikát: P, arita: 2, symbol: x_2, index: 1. Dále musí zjistit, jestli někde v derivačním stromu je stejný predikátový symbol se stejnou aritou. Pokud ano, podívá se na místo kam ukazuje jeho index pro výskyt hledaného symbolu. Tento symbol si zapamatuje a podívá se, jestli je kvantifikován. Vše je zatím obdobné jako v metodě pro generalizaci. Podstatný rozdíl je vtom, že tento symbol právě naopak nemůže být kvantifikovaný, protože v tomto pravidle musíme dosazovat term za proměnnou. Pokud ovšem kvantifikovány je, nejedná se o symbol označující term a proto ho metoda nemůže použít. Alternativní postup, pokud by k takové situaci došlo, vysvětlím na následující straně na jiném příkladu. V našem příkladu výsledný řetězec bude tento.

$$\forall y_1 P(x_2, y_1) \Rightarrow P(x_2, y_1)$$

Další krok algoritmu, nalevo $\forall$. Uplatní se pravidlo $L\forall$, to je konkretizace. Postupuje se stejným způsobem.

$$P(x_2, y_1) \Rightarrow P(x_2, y_1)$$

Nyní mají levý a pravý derivační strom v kořenu objekt označený jako predikátový symbol

a neexistuje již pravidlo, které by mohl algoritmus uplatnit. Pokud se text v těchto objektech shoduje, jedná se o axiom. Proto tímto krokem náš algoritmus končí. Formule má v Gentzenově kalkulu důkaz.

Zkusme v naší vstupní formuli $\exists x_1 \forall y_1 P(x_1, y_1) \supset \forall y_2 \exists x_2 P(x_2, y_2)$ otočit antecedent a konsekvent. Je jasné, že nám spojka implikace vytvoří formuli, která není tautologie. Pojďme se proto podívat, jak se náš algoritmus bude chovat v takovém příkladu. Na tomto příkladu vysvětlím ony alternativní postupy zmíněné v textu výše. Máme tedy zadaný sekvent reprezentovaný textovým řetězcem:

$$\forall y_1 \exists x_1 P(x_1, y_1) \Rightarrow \exists x_2 \forall y_2 P(x_2, y_2)$$

V levé formuli je v kořenu jejího derivačního stromu $\forall$ a v pravé formuli $\exists$. Podle priority má přednost pravý $\exists$. Proto jako první uplatníme pravidlo $R\exists$. To je pravidlo konkretizace. Nyní můj algoritmus nenajde symbol, který by se vázal k takovému predikátu a k takovému indexu, který není kvantifikovaný. Proto pro eliminaci kvantifikátoru musí použít jiný symbol. Zvolíme jakoukoli konstantu, která se nikde ve formuli nevyskytuje. V našem příkladě to bude symbol a.

O generování onoho symbolu se stará jiná pomocná metoda, která nabízí symboly podle abecedy. Pokud již je někde v derivačním stromu použit, nabídne nám v pořadí další. Symbol a se v našem případě nikde v derivačním stromu nevyskytuje, proto se tento symbol uplatní v onom pravidlu a dále bude platit v naší formuli jako term. Tento krok také bude korektní, a navíc nám zajistí, pokud by se nejednalo o tautologii, že by axiom nakonec nemohl vzniknout. V našem příkladu výsledný řetězec bude tento.

$$\forall y_1 \exists x_1 P(x_1, y_1) \Rightarrow \forall y_2 P(a, y_2)$$

V dalším kroku je v novém derivačního stromu nalevo kořen $\forall$ a napravo $\forall$. Podle priority má přednost pravý $\forall$. Uplatníme pravidlo $R\forall$. To je pravidlo generalizace. Symbol y_2, který je vázán kvantifikátorem, má predikát: P. Ten na levé straně má kvantifikovaný symbol y_1, podmínka je splněna, proto tento symbol použije.

$$\forall y1 \exists x_1 P(x_1, y_1) \Rightarrow P(a, y_1)$$

Další krok nalevo $\forall$ a napravo je jen predikátový symbol. Uplatní se $L\forall$, to je pravidlo konkretizace. Pomocí popsaného postupu výše bude podmínka splněna a proto se symbol y_1 může použít.

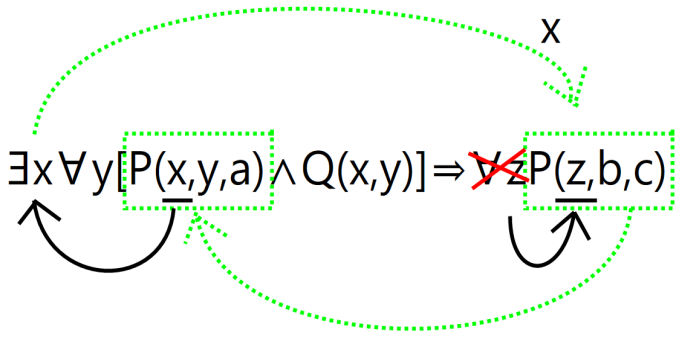
$$\exists x P(x, y_1) \Rightarrow P(a, y_1)$$

V levé formuli $\exists$ a napravo je jen predikátový symbol. Uplatní se $L\exists$, to je pravidlo generalizace. Uplatní se algoritmus pro generalizaci, v postupu popsaném výše se podmínka nesplní, proto se nechá vygenerovat konstanta, která se nikde v derivačním stromu nevyskytuje a použije se.

$$P(b, y_1) \Rightarrow P(a, y_1)$$

Nyní mají levý a pravý derivační strom v kořenu objekt označený jako predikátový symbol a neexistuje již pravidlo, které by mohl algoritmus uplatnit. Text se v těchto objektech neshoduje, nejedná se proto o axiom. Tímto krokem náš algoritmus končil. Formule nemá v Gentzenově kalkulu důkaz.

Pro lepší názornost zde ještě uvedu obrázek č. 2, na kterém princip implementovaného algoritmu pro pravidlo $R\forall$ graficky znázorňuji. V této formuli je podmínka pro pravidlo generalizace splněna, eliminujeme $\forall z$ a dosazujeme x. U všech ostatních pravidel pro kvantifikátory je tento postup analogický.



Obrázek 2: Schéma postupu algoritmu při užití pravidla $R\forall$.

Ještě uvedu příklady se všemi kombinacemi použitých kvantifikátorů, které jsme si již dokázali v předchozím textu v části: Příklady kalkulu predikátové logiky. Nyní jsou zde vyřešeny prostřednictvím našeho algoritmu, ale již bez vysvětlujícího textu, jen v podobě sekventových stromů.

Příklad 3.1.

$$R\exists\,\dfrac{L\forall\,\dfrac{P(a) \Rightarrow P(a)}{\forall x P(x) \Rightarrow P(a)}}{\forall x P(x) \Rightarrow \exists x P(x)}$$

■

Tato formule je při použití našeho algoritmu dokazatelná.

Příklad 3.2.

$$R\forall\,\dfrac{R\forall\,\dfrac{P(x) \Rightarrow P(x)}{\forall x P(x) \Rightarrow P(x)}}{\forall x P(x) \Rightarrow \forall y P(y)}$$

■

Tato formule je při použití našeho algoritmu dokazatelná.

Příklad 3.3.

$$L\exists\,\dfrac{R\exists\,\dfrac{P(y) \Rightarrow P(y)}{P(y) \Rightarrow \exists y P(y)}}{\exists x P(x) \Rightarrow \exists y P(y)}$$

■

Tato formule je při použití našeho algoritmu dokazatelná.

Příklad 3.4.

$$R\forall\,\dfrac{R\exists\,\dfrac{P(a) \Rightarrow P(x)}{\exists x P(x) \Rightarrow P(x)}}{\exists x P(x) \Rightarrow \forall y P(y)}$$

Tato formule při použití našeho algoritmu nemá důkaz, konečný segment totiž není axiom.

4 Implementace

Pro implementaci aplikace jsme vybrali platformu Windows a její programovací jazyk C#. Tento jazyk je vyvíjen společně s platformou .NET Framework. Řada inovací v jazyce C# umožňuje rychlý vývoj aplikací při zachování expresivity a elegance jazyků stylu C.

.NET je vývojová platforma pro obecné účely. Lze ji použít pro jakýkoliv typ aplikace. Tato platforma má několik klíčových funkcí, které jsou atraktivní pro mnoho vývojářů, včetně automatické správy paměti. Celá platforma je pro moderní programovaní, které umožňuje efektivně a rychle vytvářet vysoce kvalitní aplikace.

4.1 Windows Presentation Foundation

Pro grafickou vrstvu aplikace jsem zvolil (z .NET platformy) Windows Presentation Foundation, zkráceně WPF. Ta se považuje za primární API (Application Programming Interface) systému Windows Vista a vyšší. Grafický subsystém WPF poskytuje nový vzhled, nové principy přizpůsobení ovládacích prvků včetně animace 3D. WPF v praxi obsahuje dvě související programovací rozhraní C# a XAML (Extensibile Application Markup Language) založený na XML, zde se jednoduše vytvářejí kódy k obsluze událostí uživatelského vstupu. Všechna grafika funguje pomocí Direct3D knihoven, to umožňuje hardwarovou akceleraci pomocí grafické karty a pokročilejší grafické schopnosti. Na celé WPF je využívána vektorová grafika. V tomto typu grafiky jsou veškeré objekty matematicky popsány. Tyto objekty tvoří jen kotvící body se souřadnicemi, které jsou dynamicky přepočítávány. Proto není žádný problém takovou grafiku zobrazit na jakémkoliv displeji nebo dokonce plátně. Pokud je uživatelské rozhraní takové aplikace dobře navrženo, tak se perfektně přizpůsobí i poměru stran daného zobrazovacího zařízení. [9]

4.2 Uživatelské rozhraní

Základem dobrého návrhu uživatelského rozhraní aplikace je přehledné a intuitivní ovládání. Jedním ze základních doporučení při vývoji GUI (Graphical User Interface) je řídit se základními kritérii dané platformy, v našem případě Windows. Pro tuto platformu jsou sestavena základní pravidla přímo firmou Microsoft, takzvaná User Interface Guidelines. V dnešní době plné nejrůznějších dotykových a mobilních přístrojů, je tvorba takového uživatelského rozhraní ovlivněna tímto přístupem. Nejnovějším směrem v této oblasti je takzvané UWP (Univerzal Windows Platform). UWP směřuje vývojáře k tvorbě aplikací, které vypadají na všech zobrazovacích zařízeních graficky dobře a přehledně.

V našem případě je software pro výuku Gentzenova kalkulu primárně určen pro desktop. Proto jsem se zde řídil požadavky pro toto rozhraní a vývoj v UWP by byl zbytečný, ale i časově

náročný. Cílová skupina uživatelů, pro kterou je naše aplikace určena, je velice specifická a jistě se rychle přizpůsobí tomuto ovládání, proto rozhraní jen pro desktop bude dostačující.

Jako hlavní ovládací prvek aplikace jsme zvolili Ribbon (také známý pod názvem pás karet), který firma Microsoft poprvé použila v roce 2007 do svého kancelářského balíčku Office. V dnešní době je tento ovládací prvek používán ve Windows snad úplně všude, proto pro dodržení hlavních zásad User Interface Guidelines byl Ribbon jasnou volbou pro naši aplikaci. Celé GUI je ve vektorové grafice a proto jakákoliv velikost zobrazovacího zařízení by neměla být problém.

4.3 Vývoj

Celý software byl vyvíjen iterativně s čistě evolučními cykly. Dopředu byl stanoven jen rozsah projektu a cíle, které by aplikace měla splňovat. Jednotlivé komponenty celého softwaru byly psány jako samostatně spustitelné části (např.: ribbon, parser, virtuální klávesnice, pravidla pro kalkul atd.), které se jednoduše dají použít i pro tvorbu podobných aplikací. Jednotlivé komponenty byly důkladně testovány a splňují všechna kritéria dopředu stanovená vizí finálního produktu.

Obrázek 3: Hlavní okno aplikace Gentzen.

5 Uživatelská příručka k softwaru

5.1 Podpora

Software je určený primárně pro desktop, pro operační systém Windows 10.

5.2 Instalace

Pro instalaci aplikace si stačí celou složku "Gentzenův kalkul" kamkoli překopírovat a spustit program pomocí souboru Gentzen.exe.

5.3 Hlavní okno

Při prvním otevření aplikace Gentzenův kalkul se vám zobrazí hlavní okno, které je primárně nastaveno přes celý monitor. Režim dotykového ovládání se zapíná nebo vypíná malou ikonou v pravém horním rohu.

Toto okno tvoří čtyři hlavní části. V horní části je ovládací prvek ve formě pásu karet. Napravo je umístěna lišta s názvem "Asistent", ta nám slouží jako pomocný zobrazovací prostor, například pro učební texty nebo graficky znázorněné derivační stromy jednotlivých logických formulí.

Hlavní bílá plocha slouží jako prázdný list papíru pro psaní jednotlivých sekventů, a zde se i bude tvořit důkaz tautologičnosti formulí. Ve spodní části se zobrazuje virtuální klávesnice pro

zadávání logických spojek nebo kvantifikátorů. Na horní pravé části jsou umístěny šipky pro kroky zpět nebo vpřed.

5.4 Ovládací módy

Celá aplikace má dva hlavní módy, které se přepínají na první kartě. Jsou to mody "Klik"a "Text". Jak již název napovídá, buď můžeme náš důkaz tvořit jen pomocí ručně psaných textových řetězců nebo si je necháme automaticky generovat pomocí počítače, stačí jen kliknout na určité tlačítko s názvem pravidla sekventového kalkulu.

Pomocnou plochu z názvem "Asistent"si můžete kdykoli zobrazit příslušným tlačítkem a měnit její obsah pomocí záložky v pásu karet.

5.5 Jak začít

Nejprve si vložte jednu z předem nadefinovaných formulí výrokové nebo predikátové logiky 1. řádu pomocí tlačítka "Vlož formuli", které opět najdete na pásu karet v záložce "Kalkul".

Každá formule, která je předdefinovaná v seznamu, má u sebe malý barevný čtvereček. Zelená znamená, že se jedná o tauologii, červená nikoli.

5.6 Kalkul krok po kroku

1. Na kartě "Kalkul" zvolíme tlačítko "Vlož formuli"a z nabídky vybereme $\exists x A(x) \vee \exists x B \supset \exists x(A(x) \vee (B))$.
2. Pro začátek si zvolíme jednodušší mód aplikace. Přepneme v kartě "Kalkul"tlačítko "Klik".
3. V zobrazeném sekventu, klikneme myší na jeho formuli. Označí se modře.
4. V záložce "Aplikace pravidel" zvolíme pravidlo "Pravá $\supset$". Toto pravidlo se aplikuje na zatrženou formuli a počítač doplní další patro sekventového stromu.
5. Postupně si tímto způsobem budeme označovat další formule z vrchní části sekventového stromu a aplikujeme pravidla, která nám počítač bude nabízet.
6. Pozor si dejte na pravidla kvantifikátorů. Nezapomeňme, že nejprve aplikujeme pravidla generalizace a poté konkretizace. Jinak bychom mohli porušit podmínku (EVC) a axiomu bychom nikdy nedosáhli.

7. Pravidla můžeme aplikovat třeba donekonečna. Pokud jste již eliminovali všechny kvantifikátory a logické spojky, máme na výběr jen strukturální pravidla. Jistě si snadnou aplikací pravidel "LW", "RW"sestavíte axiom. Pokud ano, zdárně jste provedli důkaz tautologičnosti vybrané formule.

Celý tento proces můžeme provést i v módu "Text", jen s tím rozdílem, že další patra sekventového stromu musíme psát do nabízených text boxů. Záložka "Aplikace pravidel"v tomto módu nebude přístupná. Další patra sekventového stromu musíme doplnit sami, jen pomocí ručně psaného textu. Soupis všech pravidel kalkulu si můžete nechat zobrazit v pomocném okně "Asistent".

Pokud však vámi napsaný text nebude syntakticky v pořádku, další patro stromu se nezobrazí. Pokud syntaxe bude v pořádku a počítač nedokáže identifikovat jaké pravidlo jste užili, také vás k dalšímu patru nepustí a vedle textu se objeví dva červené otazníky. Máte možnost na tyto dva otazníky kliknout a zobrazit si tak dialogové okno s nápovědou, tam se přesně dozvíte všechny možné varianty dalšího postupu.

Nezapomeňte, že si můžete nechat zobrazovat derivační stromy jednotlivých sekventů, stačí na ně kliknout. V záložce "Okno asistent"zvolit možnost "Derivační strom". Toto vám v prvních lekcích jistě pomůže k nalezení správného pravidla. Korektně užívaná pravidla jsou ta, která odpovídají symbolům v kořeni derivačního stromu jednotlivých logických formulí. Jen se musíte orientovat, jestli je kořen napravo nebo nalevo od sekventové šipky.

6 Závěrečné zhodnocení

6.1 Možnosti dalšího rozvoje aplikace

Celá tato aplikace se i nadále může rozvíjet a doplňovat o další funkce, které budou užitečné pro studenty Gentzenova kalkulu. Rozhodně by se zde uplatnil přehlednější výstup chyb nebo možnost editovat jakýkoliv text, klidně i někde uprostřed sekventového stromu. Dají se zde doplnit funkce využívající de Morganovy zákony nebo funkce umožňující negovat formule. Užitečné by bylo i přiřazování hodnot výrokovým symbolům (true, false).

Výhledově se nemusíme držet v oblasti Gentzenova kalkulu, ale mohli bychom přepínat mezi jinými typy kalkulů. Prostředí pro zobrazování formulí máme, parser také, stačí doplnit jiné záložky s jinými pravidly. S trochou nadsázky jde vlastně ve všech kalkulech pro predikátovou logiku 1. řádu v podstatě o tentýž princip, aplikujeme pravidla kalkulu na formuli, ze které se odvozují formule další.

6.2 Programátorský pohled

Celá aplikace je psaná v *C#*, překlad takto napsaného kódu je nejprve kompilován do bytecodu, ten ale není přímo zpracovatelný procesorem, musí tedy ještě být interpretován. Proto programy psány v *C#* mají svá rychlostní omezení a nemohou dosahovat rychlostí srovnatelných s jazyky čistě kompilovanými jako je *C++*. Proto jak jsem již v textu výše uvedl, musíme používat různé subsystémy, nebo vnitřně implementované metody různých algoritmů, které jsou mnohonásobně rychlejší, než bychom je byli schopni napsat v čistém C#.

Aplikace a její implementované algoritmy nepatří mezi nejrychlejší. Celá funguje jen na dvou vláknech, jedno pro zobrazování grafiky a druhé pro výpočty a generování textových řetězců. Při dokazování dlouhých a složitých formulí si musí uživatel chvíli počkat. Jediný limit, který uživatele omezuje je čas výpočtu, protože paměťovou náročnost za vás řeší operační systém, který v případě potřeby vše přesune do virtuální paměti. Na tomto místě bych chtěl zdůraznit, že primárně byla aplikace vyvíjena jako pedagogická pomůcka a né pro nejefektivnější algoritmické řešení.

Samotný kód aplikace Gentzenův kalkul ma přibližně 10 000 řádků. Jeho implementace a ladění kódu jednomu průměrnému programátorovi zabere minimálně 400 hodin práce. Otázka nastává co je průměrný programátor? Produktivita je dána mírou zkušeností a hlavě motivací jakou k danému problému přistupuje.

V celé aplikaci nebyla použita jediná knihovna třetí strany. Vše je napsáno nově pro maximální účel splnit zadání této diplomové práce a doufám, že tento produkt bude užitečnou

pomůcku k výkladu Gentzenova kalkulu. Celý tento produkt je plně poskytnut pro potřeby VŠB, která ho může libovolně šířit a jakkoli s ním nakládat.

7 Závěr

V textu diplomové práce jsem se zabýval krátkým úvodem do matematické logiky, výčtem jednotlivých typů logických systémů typů a trochou historie. Vysvětluji zde, co jsou důkazové kalkuly a k čemu se používají. Jako hlavní cíl je seznámení čtenáře s Gentzenovým kalkulem a jeho principy dokazování. Vše vysvětluji na řešených příkladech. Rozebírám postupně jednotlivá pravidla tohoto logického kalkulu, zejména pravidla pro kvantifikátory a negaci. Hlavní přínosem je samotné sestavení univerzálního algoritmu a jeho implementace do desktopové aplikace, která bude sloužit primárně jako pedagogická pomůcka. V práci se krátce zabývám jazykem *C#* a jeho grafickým subsystémem WPF. Celou tuto práci završuje uživatelská příručka psaná jako návod pro ovládání této aplikace. Nakonec zde zmiňuji i další možnosti rozšíření tohoto softwarového produktu.

Reference

[1] BĚLOHLÁVEK, R. - VYCHODIL, V.: *Diskrétní matematika pro informatiky*. Vydavatelství UP Olomouc, 2006. s. 5-16.

[2] VIHAN, P.: *Pokroky matematiky, fyziky a astronomie*. Praha, 1992. Vol. 37, No. 5, p. 249-257.

[3] BĚLOHLÁVEK, R.: *Úvod do informatiky*. Vydavatelství UP Olomouc, 2008. s. 4-11.

[4] DUŽÍ, M.: *Logika pro informatiky (a příbuzné obory)*. Ostrava: VŠB-TU Ostrava, 2012. ISBN 9788024826622. s. 9.

[5] ŠVEJDAR. V.: *Logika - neúplnost složitost a nutnost*. Praha: Academia, 2002. ISBN 802001005X. s. 40-192.

[6] KREISEL, G.: Recenze uvedeného vydání Gentzenových spisů v: *The Journal of Philosophy*, sv. 68, č. 8, s. 238-265.

[7] KLEENE, S. C.: *Introduction to metamathematics*. New York: Ishi Press International, 2009. ISBN 9780923891572.

[8] MOLNÁR, Ľ. - ČEŠKA M. - MELICHAR B.: *Gramatiky a jazyky*. Celoštátna vysokoškolská učebnica pre elektrotechnické fakulty vysokých škôl. Bratislava: Alfa, 1987. Edícia výpočtovej techniky.

[9] PETZOLD, C.: *Mistrovství ve Windows Presentation Foundation*. Brno: Computer Press, 2008. Mistrovství. ISBN 9788025121412.